Low Level Image Processing Techniques Using the Pipeline Image Processing Engine in the Flight Telerobotic Servicer

Marilyn Nashman and Karen J. Chaconas

National Bureau of Standards, Gaithersburg, MD 20899

Abstract

This document describes the sensory processing system for the NASA/NBS Standard Reference Model (NASREM) for telerobotic control. This control system architecture has been adopted by NASA for the Flight Telerobotic Servicer. The control system is hierarchically designed and consists of three parallel systems: Task Decomposition, World Modeling and Sensory Processing. The paper will concentrate on the Sensory Processing System, and in particular will describe the image processing hardware and software used to extract features at low levels of sensory processing for tasks representative of those envisioned for the Space Station such as assembly and maintainence.

1. Introduction

The NASA/NBS Standard Reference Model (NASREM) architecture for the control system of the Flight Telerobotic Servicer defines an architecture for telerobotics based on concepts developed in other research programs. It incorporates artificial intelligence theories such as goal decomposition, hierarchical planning, model driven image analysis, blackboard systems and expert systems [1]. The multiple processes of the system are hierarchically structured. Each process is considered to be arranged vertically in a hierarchy which decomposes complex tasks into progressively simpler objectives. In addition to the vertical structure, the system is also partitioned horizontally into three sections: Task Decomposition, World Modeling, and Sensory Processing (Figure 1).

The Task Decomposition System is responsible for monitoring tasks, planning, and control servoing of the robot's manipulators, grippers, and sensors. The complexity of each function is determined by its position in the hierarchy [4, 11]. The World Model is responsible for maintaining the best estimate of the current state of the system and of the world at any given point in time. It is responsible for maintaining models of objects and structures, maps of areas and volumes, lists of objects describing features and attributes, and tables of state

variables describing the system and the environment. The Sensory Processing System is responsible for gathering sensory information from multiple instances of various sensors [8], enhancing that information [9], recognizing features, objects, and relationships between objects, and determining the correlation between observations and expectations.

Section 2 of this paper details the lower layers of the Sensory Processing System hierarchy. In Section 3, a parallel hardware system that is particularly well-suited for performing low level processing tasks is described. Section 4 explains a number of techniques employing local operations that are used to enhance data and extract features and that have been implemented on parallel hardware.

2. Sensory Processing in the NASREM Architecture

The Sensory Processing System (SPS) in the NASREM architecture [1] is designed so that data flows bidirectionally between the levels of the Sensory System and bidirectionally between the Sensory System and the World Model (Figure 1). The SPS is designed to operate in both a bottom-up (data driven) and a top-down (model driven) mode. The World Model contains both *a priori* information and updated information required to perform sensory processing tasks. At each level of the Sensory Processing hierarchy, information will be sent to the World Model. This information will be made available to the Task Decomposition module at the level in which it is needed.

The system is divided into four levels: Data Acquisition, Low Level Processing, Intermediate Level Processing and High Level Processing. This organization parallels that described in [2]. The Data Acquisition Level serves as an interface between the environment and the Sensory Processing System. It gathers raw information (readings) from each of the sensors. Depending on the complexity of the data, this information may be stored directly into the Servo Level of the World Model or used for further processing at the next level [8]. The Low Level Processor performs point-by-point operations to enhance the raw data and to perform local feature extraction. Its output is passed to the World Model at the Prim Level and/or to the Intermediate Level Processor. The Intermediate Level Processor is responsible for providing symbolic descriptions of regions, lines and surfaces that have been extracted from Low Level Processing. This data is passed both to the World Model and to the next SPS level. Lastly, the High Level Processor is responsible for interpreting and labeling the "intermediate symbolic representation" [2] and for updating the contents of the World Model with the most current knowledge about the position and orientation of objects.

3. The Parallel Image Processing Engine

The information processed by the Low Level Processor is in the form of arrays of data received from cameras, ranging sensors, or tactile array processors [9]. A typical image can consist of between 16K (128 x 128) bytes and 1M (1024 x 1024) bytes of information. Because of the large amount of data to be processed and the need to process that data as quickly as possible, most serial computers cannot meet the requirements of low level processing. Parallel computers have been developed in recent years to specifically fulfill the need of real-time processing of image data [6, 7], and although the machines differ in architectural design and implementation, they share the capability of being able to process an

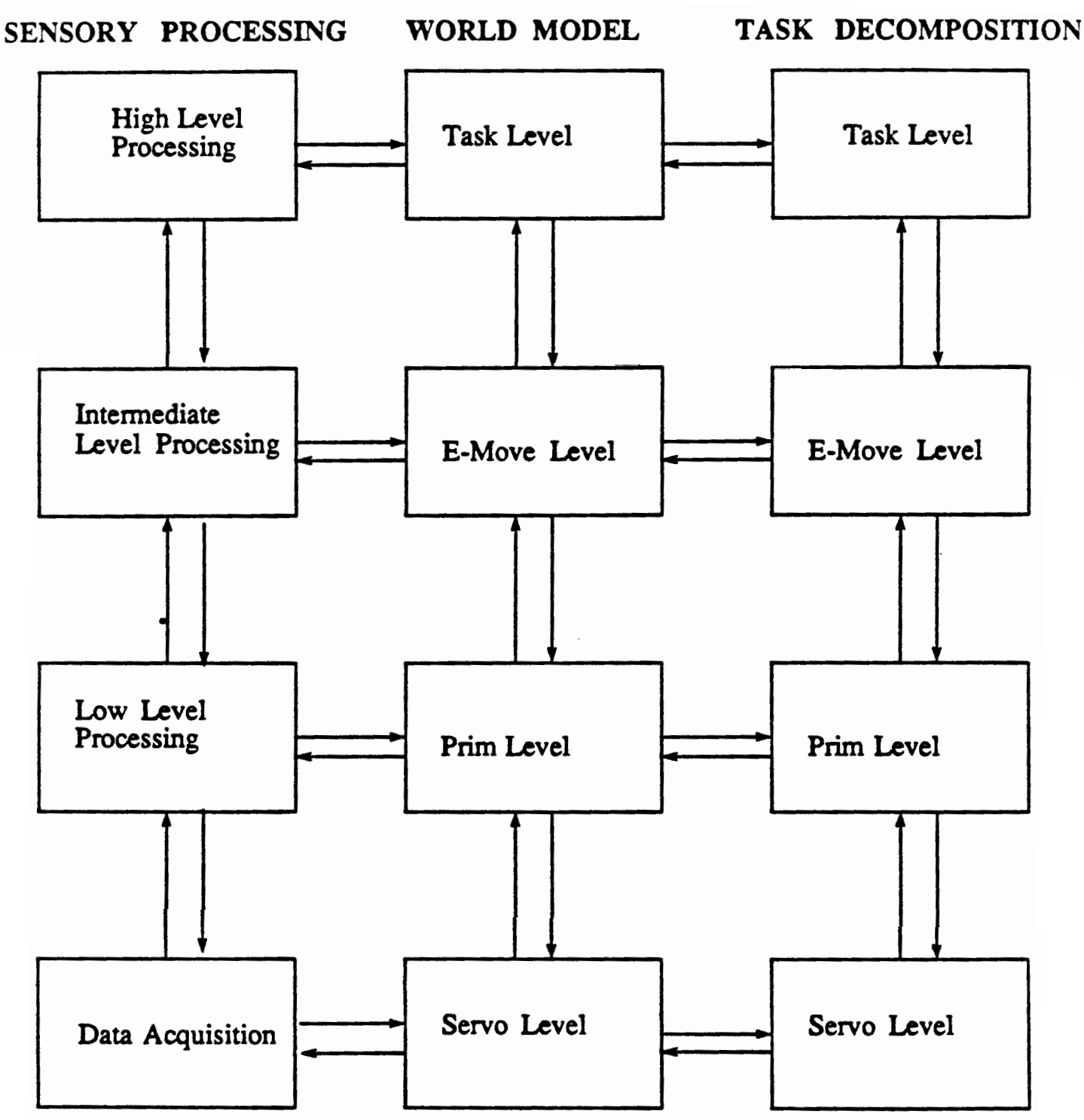


Figure 1. NASREM Hierarchy

entire image or region of an image in real-time. Parallel processing is especially applicable to low level image processing. The data structure used at this level is the image itself, a spatially indexed image of points which correspond to gray scale intensity values. All parts of the image are treated in the same way, and in general, no effort is made to distinguish between different parts of it. Local operations depend only on corresponding elements between images or on combinations of adjacent elements of an image (Figure 2). Computations tend to be simple arithmetic, algebraic, or logical operations, and typically a low number of computations per pixel is required [5]. Parallel processors are also suited to multi-resolution representations and processing techniques.

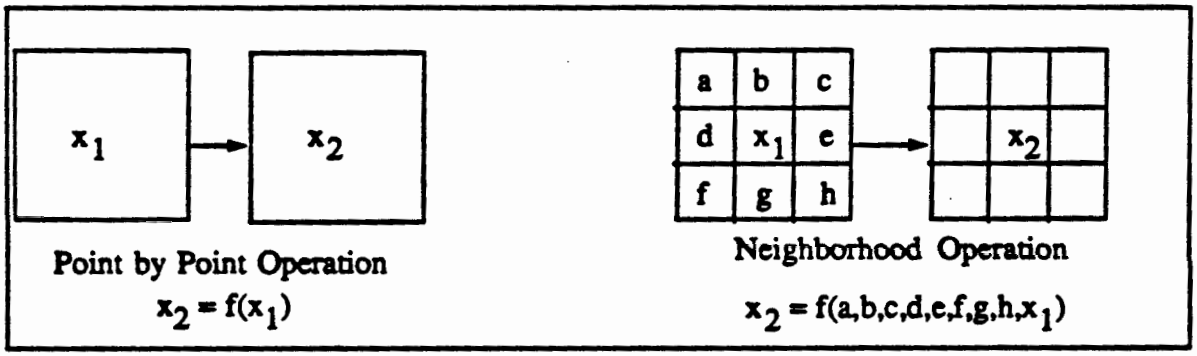


Figure 2. Local Operations

Many local data enhancement techniques can be implemented on the Pipelined Image Processing Engine (PIPE) developed at the National Bureau of Standards and manufactured by Aspex, Inc. Some features of PIPE are discussed here, but the reader is referred to [6, 7] for a more detailed description of the system. PIPE acquires its images in real-time from analog sources such as cameras, video tapes, and ranging devices, as well as digital data sources. Its output can be directed to video monitors, symbolic mapping devices, and higher level processing systems. All inputs and outputs are synchronous with the video rate of sixty fields per second.

The PIPE system is composed of up to eight identical modular processing stages, each of which contains two image buffers, look-up tables, three arithmetic logic units, and two neighborhood operators (Figure 3).

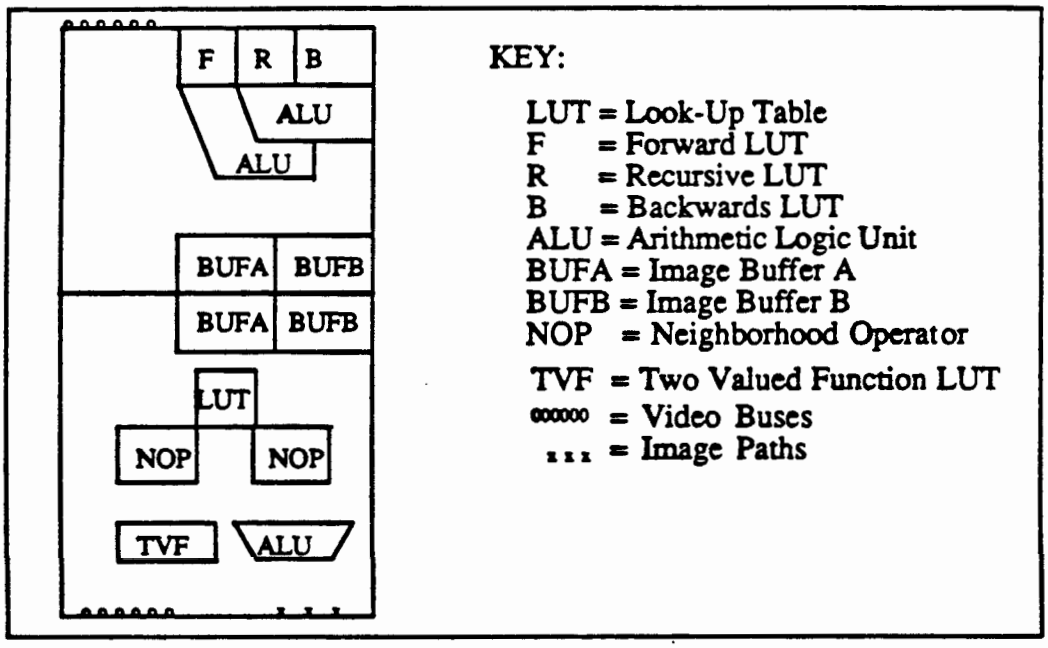


Figure 3. PIPE Modular Processing Stage

A forward path from one stage to the next allows pipelined and sequential processing. A recursive path from a stage output back to its input allows feedback and relaxation processing. A backward path from one stage to the previous stage allows for temporal operations (Figure 4) . The images in the three paths can be combined in arbitarary ways on each cycle of a PIPE program, and the chosen configuration can change on different cycles.

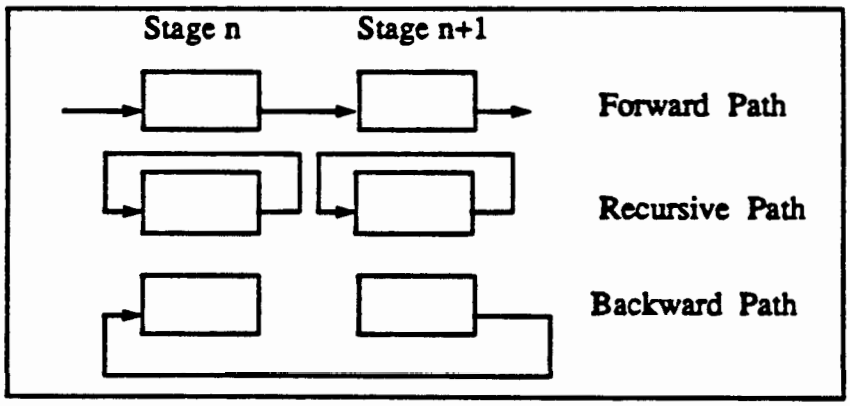


Figure 4. Data Flow Path Between PIPE Stages

In addition, six video buses allow images to be sent from any stage to any one or more stages.

Images can be processed in any combination of four ways on PIPE: point processing, spatial neighborhood processing, sequence processing or Boolean processing (Figure 5). Different processing can occur at individual pixels in the image by using a region-of-interest opera-

tor. All methods can be considered local operations.

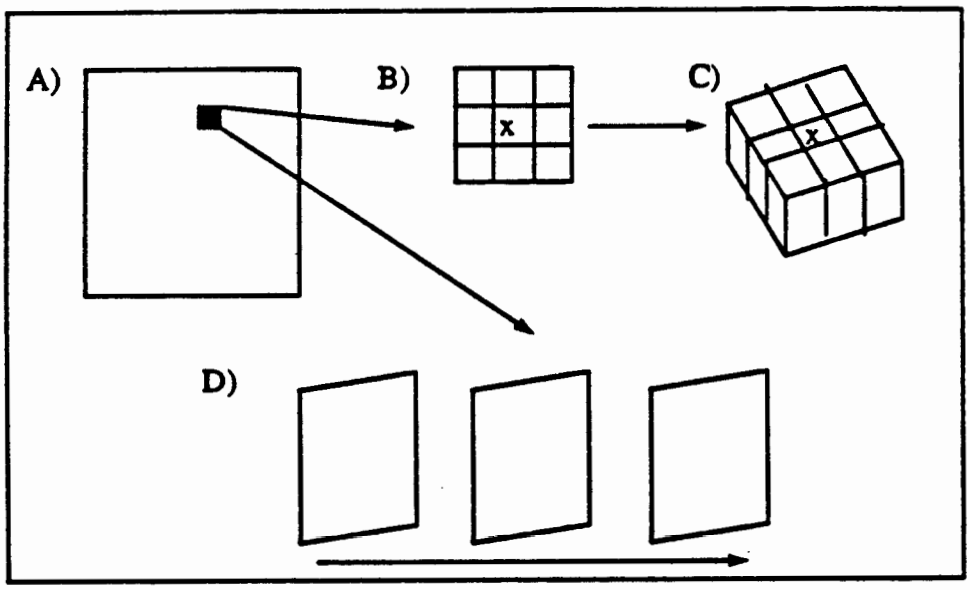


Figure 5. Processing on PIPE: (A) Point (B) Spatial (C) Boolean (D) Sequential

Point processing can be a function of either one or two input images and includes simple arithmetic and logical operations such as scaling, thresholding, converting number systems, etc. Look-up tables resident on each PIPE stage allow the user to perform more complex arithmetic operations, trigonometric operations, comparisons, rotations, etc.

PIPE can perform up to two 3 x 3 neighborhood convolutions on each stage in parallel. Both neighborhood operators operate on the same image input, but can perform different neighborhood operations. Larger neighborhood convolutions can be achieved by decomposing an odd-sized neighborhood mask into a sequence of 3 x 3 convolutions. The neighborhood operators can be either arithmetic or Boolean and are performed identically on all locations in the image unless a region-of-interest is specified. Special features are provided to prevent inaccurate computations on the image borders.

Multi-resolution pyramids can be constructed by selecting the "squeeze" or "expand" options as an image is stored or written from a buffer. In the former case, each 2 x 2 neighborhood of the input image is sampled and written to the output image resulting in an image half the resolution of the original. This process can be repeated to generate successively smaller resolution images. Expanding an image involves the opposite operation by pixel replication and generates successively larger resolution images.

Sequential processing works on a set of multiple images, e.g. sequences of images over time, a stereo pair of right and left images, or multi-resolution images. By taking advantage of the inter-stage paths, images can be combined, compared, sampled or differenced to extract the desired application dependent information.

When performing Boolean processing, each pixel of information is considered to be composed of eight independent bit planes which are operated upon simultaneously. The neighborhood operators can be applied in a Boolean mode, where the output is the combination of

the 3 x 3 neighborhood using local operations on each of the eight bit planes.

PIPE programs are written on a host computer using a software package which is an iconic representation of the hardware to generate microcode. The microcode instructions are downloaded to PIPE, where they are resident during program execution. A software development tool, ASPIPE, allows the user to code the spatial and temporal flow of the data through the hardware and to allocate the look-up tables and PIPE resources to be used. Programs can be edited, saved, compiled, executed, and debugged in this environment. In addition, ASPIPE generates a sequencer file that specifies which micro-operation is executing at each time-cycle. This sequencer also controls branching and looping among microcode instructions during execution.

A hardware interface between PIPE and a high level processor (HLP) has been developed and software has been written to support this interface. In this manner, the results of low level vision tasks are transferred to a serial computer which can perform high level vision tasks of image analysis, recognition, and general decision making which require global information. Since the interface is bidirectional, the HLP can download images or look-up tables directly to any buffer or table on any selected piece of PIPE hardware. In addition, the HLP can select PIPE algorithms by manipulating the PIPE sequencer.

4. Low Level Image Processing Algorithms

Figure 6 is a picture of a truss node suggested for use in assembly of the NASA Space Station. The sockets are attached to the node in various configurations, but the world model has knowledge of the geometry of each instance of any assembly. The appearance of the truss node presents a difficult problem for computer vision: the part is machined of a smooth, highly reflective metal, and the curvature of the node increases the difficulty of obtaining satisfactory information with standard image processing techniques.

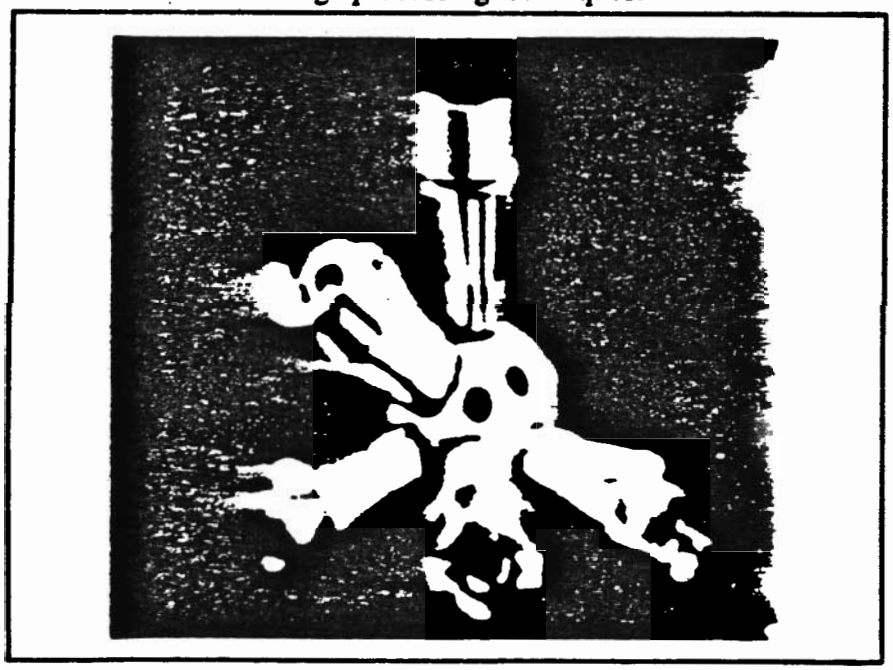


Figure 6. Truss Node Assembly

Binary thresholding of the image fails because of the specularity of the node. Connected component algorithms which segment an image into distinct objects and compute statistical information relative to each object fail because the node is improperly segmented due to highlight and shadow effects. Edge extraction routines provide extraneous information because highlights are falsely interpreted as edges. Figure 7 illustrates the "edges" found in the truss node assembly using a non-maxima suppression algorithm. .

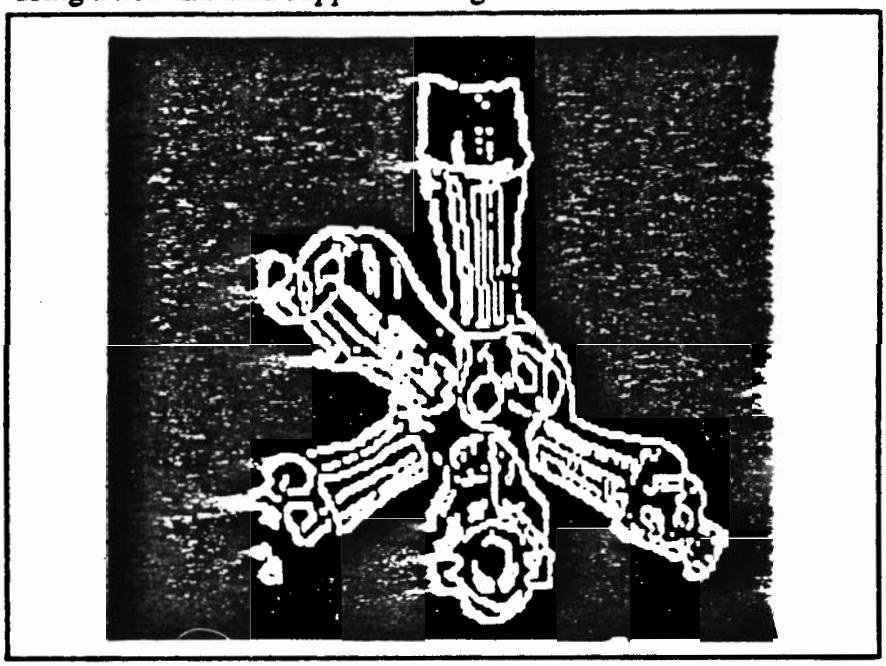


Figure 7. Truss Node Edge Image

To overcome these obstacles, an algorithm was developed on PIPE which makes use of standard edge extraction techniques, image smoothing, and multi-resolution processing. The goal of this algorithm is to provide a connected edge image of the truss node assembly which can be used as input to a connected component algorithm.

The first operation applied in this algorithm involves extracting edges in the full resolution image. A Sobel operator [10] is applied to the image using PIPE's neighborhood operator to extract the x and y gradients at each pixel in the image (Figure 8).

X Gradient Operator

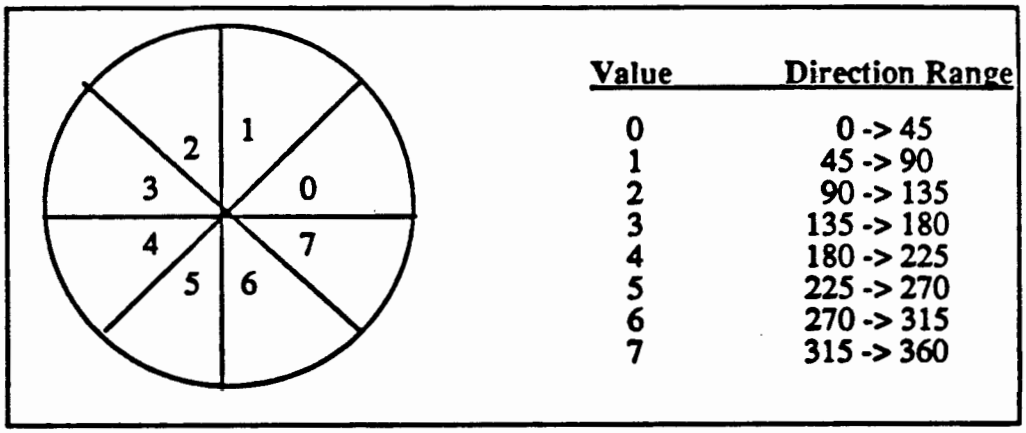
-1	0	1
-2	0	2
-1	0	1

Y Gradient Operator

1	2	1
0	0	0
-1	-2	-1

Figure 8. Sobel Operator

The magnitude and direction of each edge point are then computed using two-valued function look-up tables. By thresholding the direction image with the magnitude image to remove weak edges, a three pixel wide, binary edge is obtained . In order to thin the edge image, a non-maxima suppression algorithm is applied. This operation involves quantizing the directions of all edge points into one of eight values (Figure 9). The output of this quantization



Value	Direction Range
0	0 -> 45
1	45 -> 90
2	90 -> 135
3	135 -> 180
4	180 -> 225
5	225 -> 270
6	270 -> 315
7	315 -> 360

Figure 9. Quantization of Direction Image

is stored in a buffer which is used to determine in which direction to thin the corresponding pixel in the edge image. In this manner, different 3 x 3 masks can be applied to the image depending on the direction of the edge, and all edge points that are not maximum in the gradient direction are eliminated (Figure 7).

In order to remove the extraneous information in the thinned edge image, multi-resolution processing is used. The image is first smoothed using a Gaussian operator [10], and then it is sampled such that each 2 x 2 neighborhood of the original image is averaged to produce one pixel at the next higher level of resolution (Figure 10). The reverse operation is then applied to the smoothed sampled image; it is expanded back to a 256 x 256 image using pixel replication.

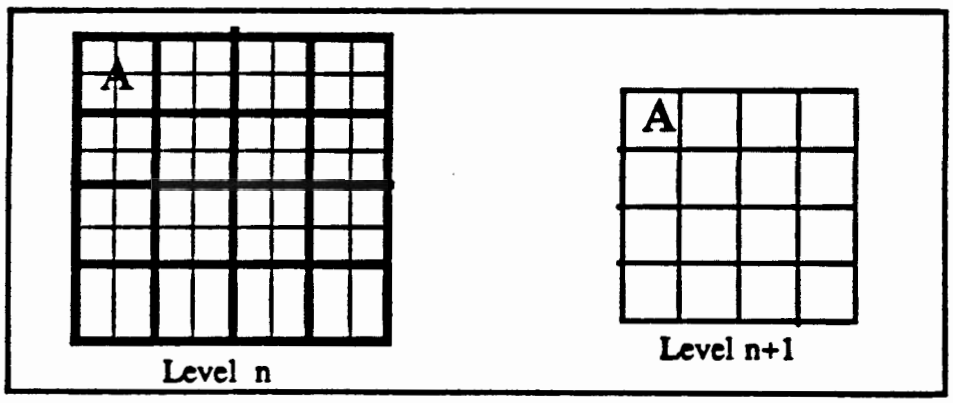


Figure 10. Forming Levels of a Multi-Resolution Pyramid

The result of these operations is shown in Figure 11. The false edges caused by the specu-

larity have been removed and all portions of the truss assembly are connected. Reapplying the Sobel operator to Figure 11 results in a connected edge image (Figure 12), and applying a shrinking algorithm results in a connected, thinned edge image (Figure 13).

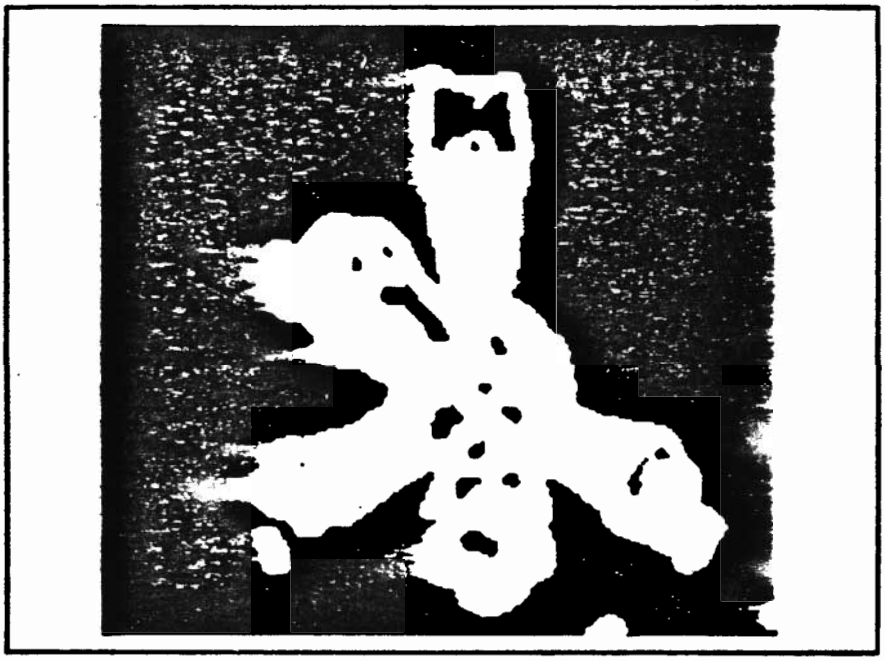


Figure 11. Result of Multi-Resolution Processing

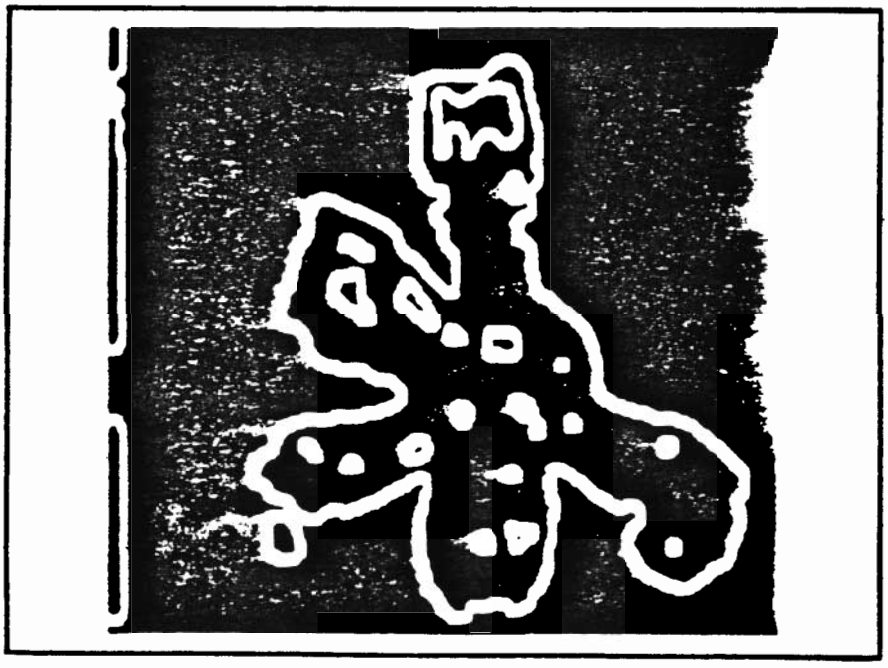


Figure 12. Sobel Edge Image

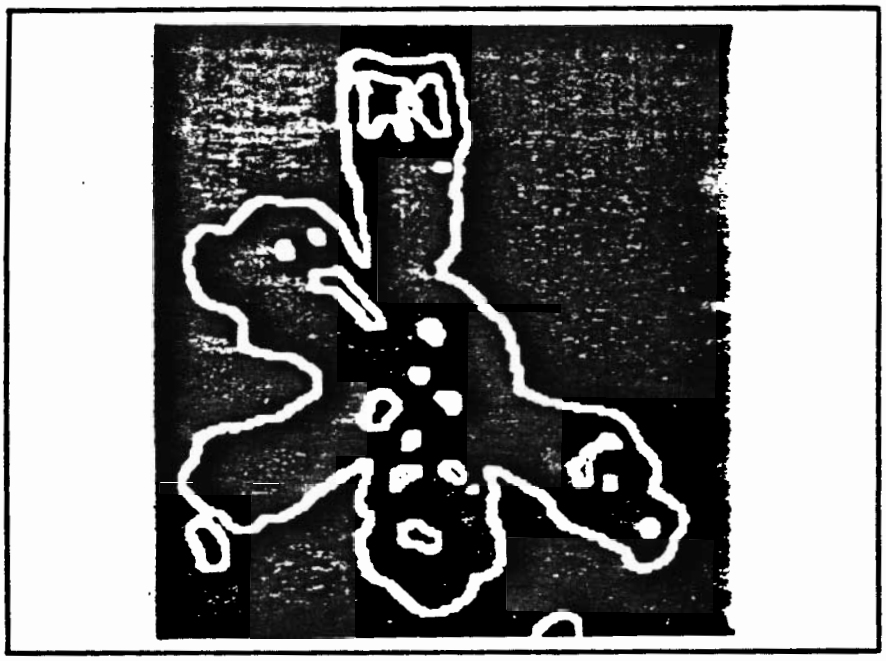


Figure 13. Thinned Edge Image

Using the hardware interface between PIPE and the HLP, the thinned edge image is transferred to the HLP for additional processing to obtain global information. In particular, the area of the node, its centroid, and its orientation are computed using the (p+q)th order moments defined in [12]:

$$m_{pq} = \int\int x^p y^q f(x,y)\, dx\, dy$$

where f(x,y) = 1 for all edge points and f(x,y) = 0 for all non-edge points. The centroid of an object is defined as :

$$x_c = m_{10} / m_{00}\,,\; y_c = m_{01} / m_{00}$$

where m_{00} is the area of the object, and the orientation is defined as :

$$\theta = .5 \tan^{-1} [\, 2\,(m_{00}\, m_{11} - m_{10}\, m_{01}) / ((m_{00}\, m_{20} - m_{10}^2) - (m_{00}\, m_{02} - m_{01}^2))].$$

The locations of corners of an object provide useful information in that they support the calculation of the orientation of an object. Given the model of an object, the viewing position can be determined by knowing which corners are visible.

Corners can be defined as locations where adjacent edge segments have high rates of curvature. These rates of curvature can be measured over small distances, yielding local corners. As the distance becomes larger, more global corners are found. To detect global corners, it is useful to use a lower resolution image, since a large area in the high resolution image maps to a relatively smaller area in the low resolution image (see Figure 10). A corner detection algorithm was implemented on the PIPE using these concepts.

Initially, an image of the truss node (see Figure 6) was used to generate successively lower resolution versions of the same image. The image was sampled so that only every other pixel on every other row was used to produce an image at the next resolution. From a 256 x 256 image, images were created of sizes 128 x 128, 64 x 64, 32 x 32, and 16 x 16. Using the low resolution image, a Sobel edge operator was applied to compute edge magnitude. Figure 14 is a picture of the edge image at this low resolution thresholded to indicate where edges resulted from high changes in contrast. Next, four Boolean neighborhood operations were computed on this binary edge image to test for the presence of eight types of corners (see Figure 15). The responses from the corner masks were combined and then expanded back to full resolution using pixel replication. The results are shown superimposed on the grey scale image of the truss node, where the corners were detected on the 16 x 16 level (see Figure 16) and on the 32 x 32 level (see Figure 17). As is expected, there are more responses obtained at the 32 x 32 level of resolution than at the 16 x 16 level. This is caused by the fact that the local operators are applied over a smaller distance, thereby detecting more local corners.

The quality and accuracy of the corners detected depend largely on the level of resolution at which they were extracted. The margin of error of the corner position is produced as a result of the way in which images are reduced and expanded on the PIPE. As an image buffer is reduced in resolution, pixels are sampled in every other row and in every other column. The result is always placed in the same corner of a 2 x 2 neighborhood. Expansion of an image involves the replication of pixels in a 2 x 2 neighborhood. Thus a corner point in the 16 x 16 image represents 16 pixels in the 256 x 256 image, any one of which can be a true corner point.

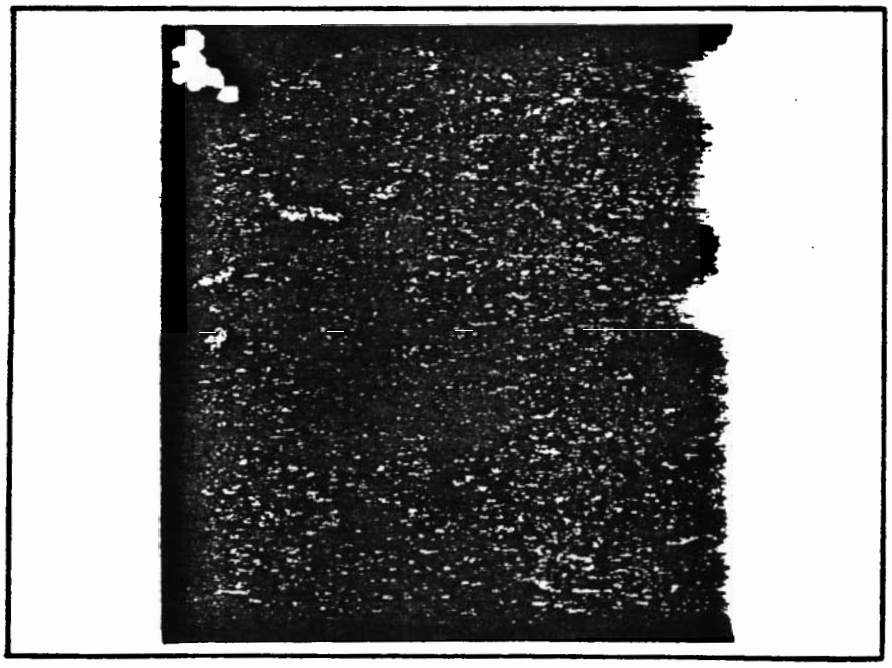


Figure 14. Low Resolution Image of Truss Node

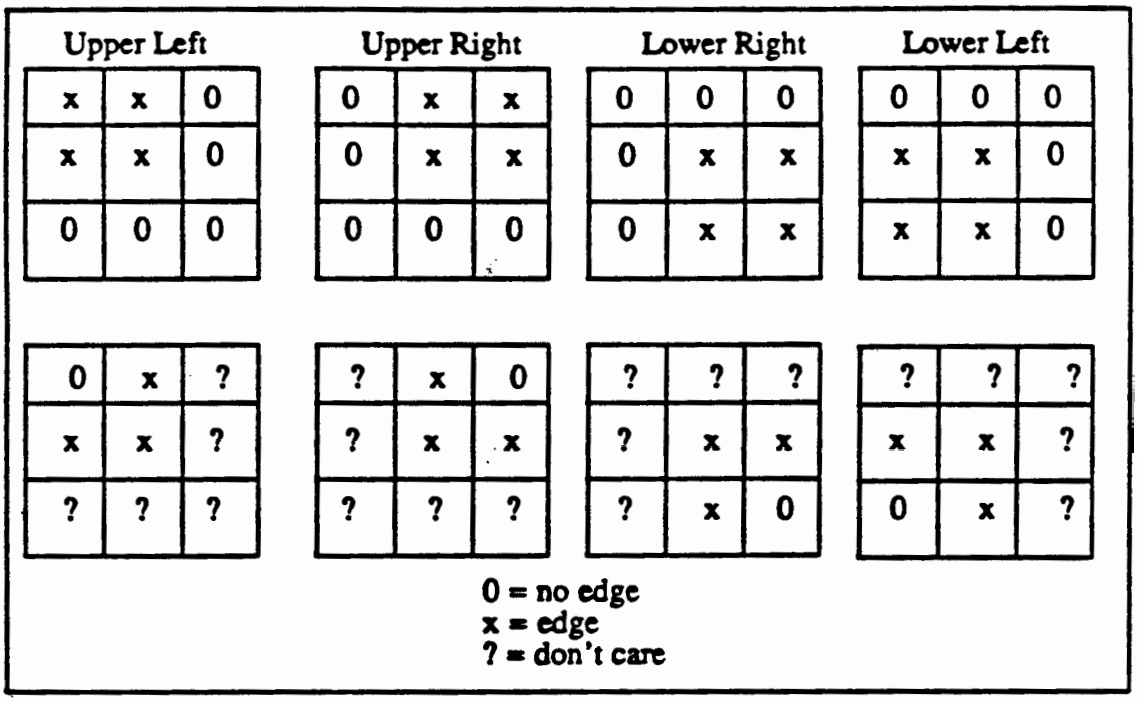


Figure 15. Corner Detection Masks

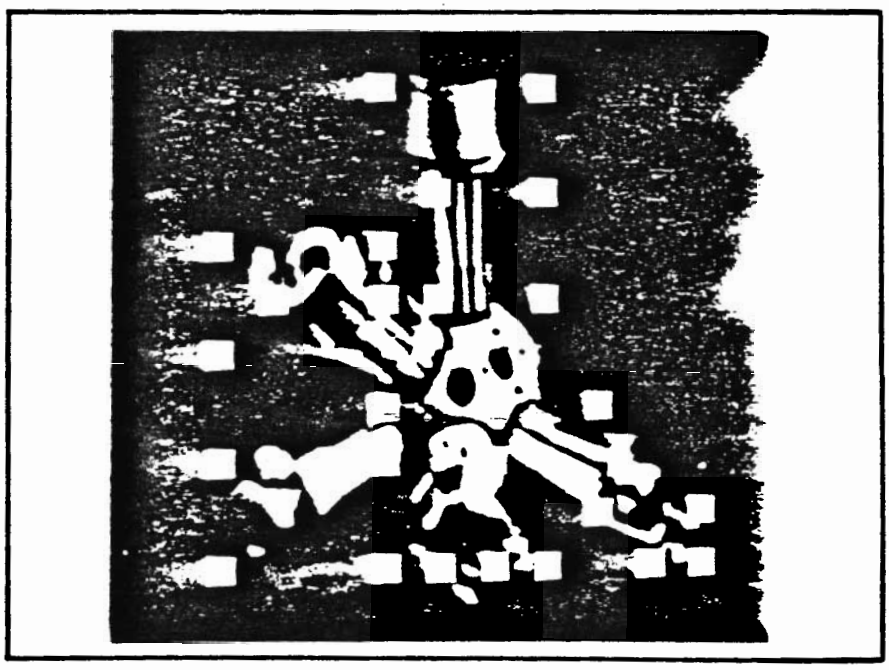


Figure 16. Corners Detected on 16 x 16 Level

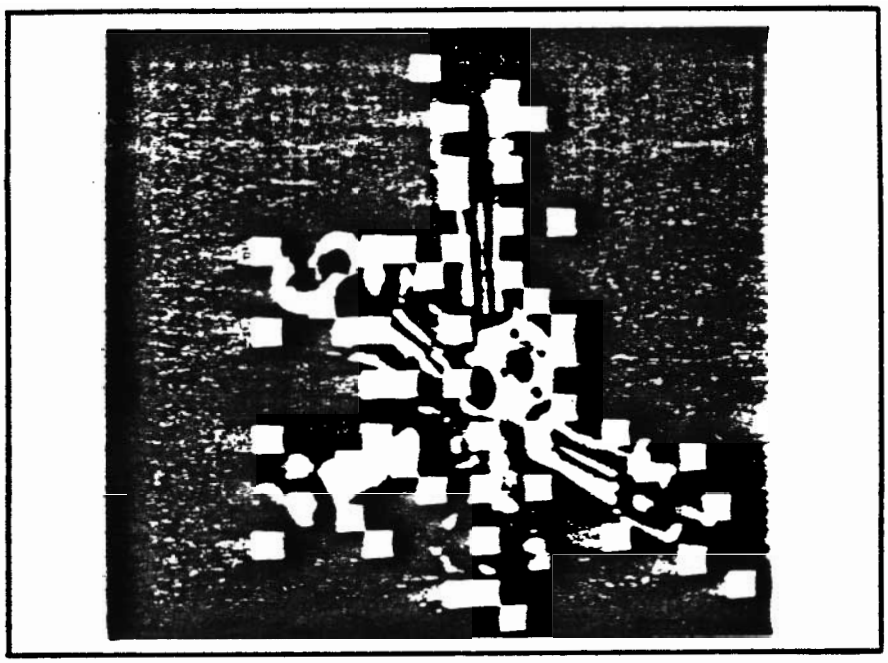


Figure 17. Corners Detected on 32 x 32 Level

5. Conclusion

A Flight Telerobotic Servicer will be used to assist the astronaut in the construction and maintainance of the NASA Space Station. NASREM, the hierarchical control system developed at the National Bureau of Standards (NBS), has been chosen by NASA as the computing architecture for this project. The sensory processing portion of this control scheme involves, at the low level, the preprocessing and enhancement of large arrays of data that have been gathered from external sensors. Feature extraction from these arrays is often more accurate if the data is preprocessed to remove the effects of noise and variable environmental conditions such as lighting.

Using a truss node, a structure which will be used in the Space Station, the PIPE was able to produce meaningful information which will be used by other processes in the control scheme. By using a combination of edge extraction techniques, image smoothing, and multi-resolution processing, image processing problems presented by the specularity of the truss node were overcome. A PIPE program is able to produce a thinned, connected edge image approximately every 1/10th of a second. A second PIPE program has been used to extract corners or areas of high curvature at update rates of 1/4th of a second. These results enable higher sensory levels to compute the position and orientation of the node in space.

More effort is required to bind corners detected at low levels of resolution with their true position in the full resolution image. This corner localization can be accomplished by adjusting the corner positions on each level of resolution during expansion instead of only at the lowest level. In addition, both algorithms can be enhanced by the removal of spurious edge points in the image.

Acknowledgement

The authors wish to thank Tsai-Hong Hong for her helpful suggestions and evaluations of algorithms implemented on the PIPE.

References

[1] Albus, J.S., McCain, H.G., Lumia, R., "NASA/NBS Standard Reference Model Telerobot Control System Architecture (NASREM)", NASA Document SS-GSFC-0027, December 4, 1986.

[2] Arkin, R.C., Riseman, E.M., Hanson, A.R., "AuRA: An Architecture for Vision-Based Robot Navigation", Proceedings: Image Understanding Workshop, February, 1987.

[3] Aspex, Inc., "PIPE--An Introduction to the PIPE System", 1987.

[4] Fiala, J. , "Manipulator Servo Level Task Decomposition", Doc. ICG #002, NBS Internal Report, 1987.

[5] Gross,T., Lam,M., Webb,J.,"WARP As A Machine for Low Level Vision", IEEE Conference on Robotics and Automation, 1985.

[6] Kent,E., Shneier,M., Lumia,R., "PIPE-Pipelined Image Processing Engine", Journal of Parallel and Distributed Computing, 1984.

[7] Lumia, R., Shneier, M., Kent, E., "A Real-Time Iconic Image Processor",NBSIR, 1984.

[8] Nashman, M., Chaconas, K., "Sensory Processing System, Data Acquisition Level", ICG #005, NBS Internal Report, 1987.

[9] Nashman, M., Chaconas, K., "Sensory Processing System, Low Level Processing Stage", ICG #012, NBS Internal Report, 1988.

[10] Rosenfeld, A., Kak, A., "Digital Picture Processing", Volume 1 Second Edition, Academic Press, 1982.

[11] Wavering, A., "Manipulator Primitive Level Task Decomposition", ICG #003, NBS Internal Report, 1987.

[12] Wilf, J.M., Cunningham, R.T., "Computing Region Moments from Boundry Representations", JPL Publication 79-49, November, 1979.